# EPL221: Quick Reference Card

## Linux Commands:

| Command | Example | Comments |
|---|---|---|
| `pwd` | | Shows the current directory |
| `ls` | ls -lsa | List of files in the current directory |
| `cat` | cat example1.s | Display contents of file |
| `man` | man echo | Display command Manual |
| `grep` | grep Hello example1.s | Searches text data |
| `ps` | ps aux | Lists Processes |
| `mv oldFIleName.s newFileName.s` | | Move/Rename from oldFileName.s to newFileName.s |
| `cat quiz1.fl` | | Εκτύπωση του αρχείου στην οθόνη και να ελέγξουμε ότι περιέχει τις αλλαγές που κάναμε στο file. |
| `[Tab]` | | Automatic Completion (Εύκολος τρόπος για να γράψουμε σωστά το όνομα ενός αρχείου ή μιας εντολής) |
| `[arrow up]` | | Μας δίνει τις προηγούμενες εντολές που έχουμε εκτελέσει. |

## vi Commands:

| Command | Example | Comments |
|---|---|---|
| `:set number` `:set num` | | Show the line numbers on the left. |
| `vi filename +line` | **vi example.s +5** | Open vi and takes you to line 5 |
| `;/Keyword` | :/Hello | Search for word in vi and less |
| `vi quiz1.fl` | | Ανοίγουμε το αρχείο quiz1.fl με τον vi |
| `i` | | Edit Mode for vi |
| `[esc]` | | Πατώντας το κουμπί esc θα σας πάρει πίσω στο command mode. |
| `:` | | Πατώντας το κουμπί : θα σας δώσει την δυνατότητα να δώσετε εντολές. |
| `wq` | | Write file and Quit (w = write, q = quit) |

**gcc Commands:**

| Command | Example | Comments |
|---|---|---|
| `gcc -g filename.s` | **gcc -g example1.s** | Compiles for Debugging |
| `gcc -g filename.s -o file1` | **gcc -g filename.s -o file1** | Compiles **filename.s** and saves the executable as **file1** |

**gdb Commands:**

| Command | Example | Comments |
|---|---|---|
| `b function` | b main | Set a Breakpoint at the start of a function (main in this case) |
| `r` | | Run the Program |
| `Ctrl+w Crtt+2` | | Switch to 2 screen layout. |
| `info regs` | | Prints the content of all registers |
| `p/x $register` | p/x $pc | Prints in hex the content of a specific register |
| `bt` | | Back Trace function calls. |
| `x/1s 0x4109ec` | | Prints a String in the Memory. Use objdump to discover where the String is stored in .data.<br>(gdb) x/1s 0x4109ec<br>0x4109ec: "Hello Wolrd!!!\n" |
| `x/4b 0x4109f4` | | Prints 4 bytes starting from the address 0x4109f4<br>(gdb) x 0x4109f4<br>0x4109f4: 0x1234abcd<br>(gdb) **x/4b 0x4109f4**<br>0x4109f4: 0xcd 0xab 0x34 0x12 |

**objdump Commands:**

| Command | Example | Comments |
|---|---|---|
| **objdump -d executable \| less** | objdump -d ./a.out \| less | Disassembles the executable and sends it to less for viewing |
| **objdump -d .executable -j .text \| less** | objdump -d ./a.out -j .text \| less | Sends the .text disassembly to less. You can search for main function using /main in less. |
| **objdump -sj .data executable** | objdump -s -j .data ./a.out | Print the .data<br>objdump -sj .data ./a.out<br><br>./a.out: file format elf64-littleaarch64<br><br>Contents of section .data:<br>4109e8 00000000 48656c6c 6f20576f 6c726421 **....Hello Wolrd!**<br>4109f8 21210a00 **!!..** |
| **objdump -dsj .data .executable** | objdump -dsj .data ./a.out | objdump -dsj .data ./a.out<br><br>./a.out: file format elf64-littleaarch64<br><br>Contents of section .data:<br>4109e8 00000000 48656c6c 6f20576f 6c726421 ....Hello Wolrd!<br>4109f8 21210a00 !!..<br><br>Disassembly of section .data:<br><br>00000000004109e8 <__data_start>:<br>4109e8: 00000000 .word 0x00000000<br><br>00000000004109ec **<Hello_Label>:**<br>4109ec: 6c6c6548 .word 0x6c6c6548<br>4109f0: 6f57206f .word 0x6f57206f<br>4109f4: 2164726c .word 0x2164726c<br>4109f8: 000a2121 .word 0x000a2121 |

**hexdump Commands:**

| Command | Example | Comments |
|---|---|---|
| **hexdump -C binaryFile \| less** | hexdump -C ./a.out \| less | Displays the content of a binary file in hex and ascii format. |

# AAPCS64: Role of integer registers

| Register | Alternative name | Role |
|---|---|---|
| R0 | | Return value (for integers and pointers) |
| R0 … R7 | | Arguments in function calls (for integers and pointers) |
| R8 | | Indirect result location register. Used in C++ for returning non-trivial objects (set by the caller). |
| R9 … R15 | | Temporary registers (trashed across calls) |
| R16, R17 | IP0, IP1 | The intra-procedure-call temporary registers. The linker may use these in PLT code. Can be used as temporary registers between calls |
| R18 | | Platform register |
| R19 … R28 | | Callee-saved registers: register preserved across calls |
| R29 | FP | Frame pointer. Copy of SP before function stack allocation |
| R30 | LR | Link register. BL and BLR instructions save return address in it |
| SP | | Stack pointer |

The full list of condition codes is as follows:

| Encoding | Name (& alias) | Meaning (integer) | Meaning (floating point) | Flags |
|---|---|---|---|---|
| 0000 | EQ | Equal | Equal | Z==1 |
| 0001 | NE | Not equal | Not equal, or unordered | Z==0 |
| 0010 | HS (CS) | Unsigned higher or same (Carry set) | Greater than, equal, or unordered | C==1 |
| 0011 | LO (CC) | Unsigned lower (Carry clear) | Less than | C==0 |
| 0100 | MI | Minus (negative) | Less than | N==1 |
| 0101 | PL | Plus (positive or zero) | Greater than, equal, or unordered | N==0 |
| 0110 | VS | Overflow set | Unordered | V==1 |
| 0111 | VC | Overflow clear | Ordered | V==0 |
| 1000 | HI | Unsigned higher | Greater than, or unordered | C==1 && Z==0 |
| 1001 | LS | Unsigned lower or same | Less than or equal | !(C==1 && Z==0) |
| 1010 | GE | Signed greater than or equal | Greater than or equal | N==V |
| 1011 | LT | Signed less than | Less than or unordered | N!=V |
| 1100 | GT | Signed greater than | Greater than | Z==0 && N==V |
| 1101 | LE | Signed less than or equal | Less than, equal, or unordered | !(Z==0 && N==V) |
| 1110 | AL | Always | Always | Any |
| 1111 | NV[†] | | | |